Lab Report SS24

Mridul Krishnan , Muhammad Ibrahim Afsar Khan , Mansi Pandit

October 4, 2024

Abstract

This lab report details the development and evaluation of a self-supervised learning (SSL) approach for depth and motion estimation in automotive scenes. The goal is to estimate depth, camera motion, and optical flow in an end-to-end manner, using self-supervised training without the need for costly ground truth annotations. The proposed model includes a shared convolutional feature extractor, per-task decoders for depth, ego-motion, and optical flow estimation, and is trained for novel view synthesis by predicting the camera motion and optical flow from sequential frames. Training strategies incorporate photometric and regularization loss functions to ensure smoothness and consistency. The Cityscapes dataset is utilized for training and evaluation, with metrics such as mean squared error (MSE) and root mean squared error (RMSE) used to quantify performance. The project aims to explore improvements in model architecture, training schedules, and loss functions to enhance accuracy in depth and motion estimation tasks.

1 Introduction

Depth and motion estimation are crucial components of modern computer vision systems, particularly in the context of autonomous driving. These tasks, while straightforward for humans, are complex and computationally demanding for machines. Accurate estimation of depth and motion is essential for scene understanding, object detection, and navigation in automotive applications. Traditional methods for obtaining depth data rely on expensive sensors, and the ground truth annotations required for training are both costly and time-consuming to acquire.

To address these challenges, this project explores a self-supervised learning (SSL) approach to estimate depth, camera motion, and optical flow directly from sequential video frames. The proposed model is designed to operate end-to-end, leveraging shared encoders and dedicated decoders for each task. By training the model to perform novel view synthesis, we enable the system to learn these representations without relying on costly ground truth.

The use of the Cityscapes dataset provides an excellent basis for evaluating our approach in real urban driving scenarios. We utilize computed depth and

optical flow maps, obtained from pre-trained models, to validate our model's performance. Our training strategy includes photometric and regularization loss functions to ensure smooth and consistent estimations across scenes.

This project aims to advance the state of SSL-based depth and motion estimation by experimenting with different encoder backbones, training strategies, and architectural variations. Our findings contribute to more efficient, cost-effective approaches to depth and motion perception, which are fundamental for the advancement of autonomous vehicle technologies.

2 Methodology

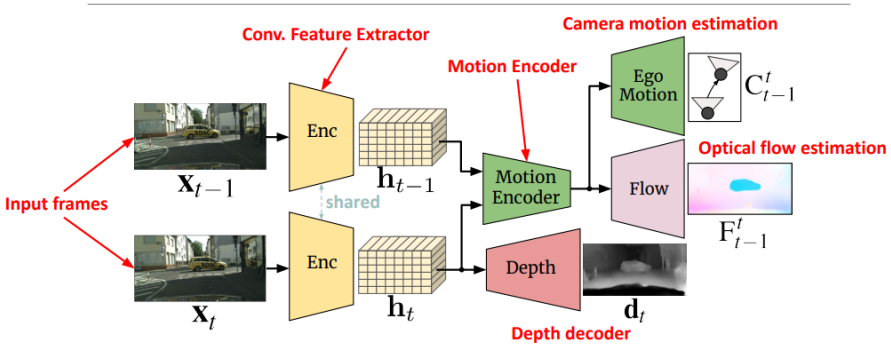

Figure 1: An example of depth and motion estimation using the proposed model.

The proposed methodology for depth and motion estimation consists of several key components, as shown in Figure 1. The process can be broken down into the following steps:

2.1 Input Frames and Feature Extraction

The model takes two consecutive input frames, x_{t-1} and x_t, which represent images captured at different time steps. These frames are passed through a convolutional feature extractor, which in our case is a pretrained ResNet18 [3], denoted as *Enc*, to produce feature maps h_{t-1} and h_t:

$$h_{t-1} = \text{Enc}(x_{t-1}), \quad h_t = \text{Enc}(x_t) \tag{1}$$

The feature extractor is shared between both input frames to ensure that the features are consistent across time.

2.2 Motion Encoder and Camera Motion Estimation

The extracted feature maps h_{t-1} and h_t are concatenated and passed to a motion encoder to estimate the camera motion, also known as ego-motion.

2.2.1 Motion Encoder

The motion encoder produces the features required for estimating the relative transformation between frames:

$$h_{\text{motion}} = \text{MotionEncoder}([h_{t-1}, h_t]) \tag{2}$$

The motion encoder uses a pre-trained ResNet encoder, which is used to extract high-level feature maps from two images: the current frame and the target frame (the previous frame). These feature maps, denoted as h_{t-1} and h_t are extracted from both frames using the same encoder network. Afterward, the two feature maps are concatenated along the channel dimension, resulting in a combined feature map with twice the number of channels.

Once the feature maps are concatenated, the resulting feature tensor is passed through three convolutional layers, each followed by batch normalization and ReLU activation. These layers progressively reduce the channel dimensions from 1024 to 128, capturing meaningful motion information between the two frames. The final output, a tensor with 128 channels, represents the learned motion features, which can be used for tasks such as motion estimation, action recognition, or temporal analysis in video data. The batch normalization layers help stabilize the learning process, while the ReLU activations introduce non-linearity to capture more complex relationships in the motion patterns.

2.2.2 Ego Motion Estimation

The camera motion estimation module then outputs the camera transformation C_{t-1}^t, which includes the translation and rotation (Euler angles) of the camera between time steps:

$$C_{t-1}^t = f_{\text{ego}}(h_{\text{motion}}) \tag{3}$$

The implemented model follows the implementation from MCDS-VSS [5]. The model uses the above mentioned motion encoder to extract motion-related features from the input frames. These features are then passed into a decoder processes these features to predict the pose. After receiving the motion features, the decoder then processes these motion features through a series of convolutional layers. It reduces the channel dimension from 128 to 64, then to 32, and finally outputs a tensor with 6 channels, corresponding to the six components of the pose (3 for rotation and 3 for translation). After the convolutions, global average pooling is applied across the height and width dimensions to reduce the spatial size, producing a 6D vector that represents the predicted pose. This vector encodes the camera's or object's movement between the two frames.

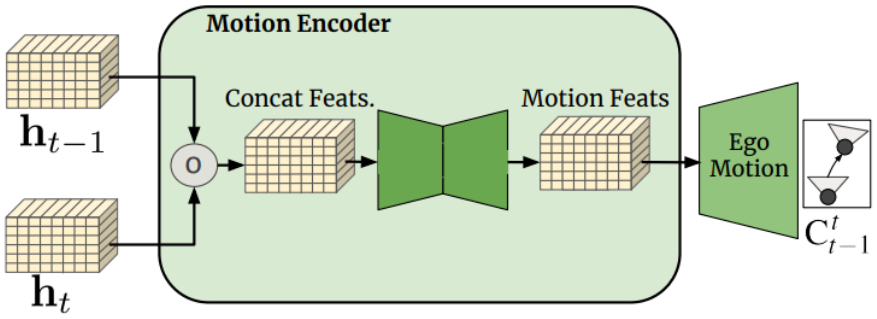

Figure 2: Ego Motion Model.

We used the following photometric loss function for the ego motion estimation:

$$\mathcal{L}_{\text{Photo}} = \frac{\alpha}{2}(1 - \text{SSIM}(\hat{x}^{\text{ego}}, x_t)) + (1 - \alpha)|\hat{x}^{\text{ego}} - x_t|_1 \tag{4}$$

2.3 Depth Estimation

The implemented model for depth estimation of a sequence of images is used to analyze the movement of objects in the sequence and estimation their distance/depth from the focal point, i.e. the camera. The input to the model is a single frame of a sequence which is first encoded using a pretrained ResNet18 [3] model. The result of the encoder is an encoded feature space that is then provided as input to the decoder of the depth estimation model. The decode is inspired by MonoDepth2 [2].

The decoder is designed to predict depth maps from the feature maps. Its architecture consists of a series of up-sampling blocks, which progressively refine and upscale the lower-resolution feature maps to recover spatial details. Each up-sampling block includes an up-sampling operation followed by a convolutional layer, batch normalization, and a ReLU activation. The model's up-sampling path is structured in five layers, with each layer halving the number of channels from the previous one.

At different scales of resolution, the model uses specific output convolutional layers to produce depth predictions, creating depth maps at multiple scales. After convolution, the outputs are passed through a sigmoid activation function to normalize the predicted depth values between 0 and 1. This allows the model to generate depth maps at different levels of granularity. The output of the model is a list of four depth maps at progressively increasing resolutions, providing predictions at various scales for more flexible and accurate depth estimation. For all results the first output scale is used.

The loss function used for the depth estimation is regularization loss:

$$\mathcal{L}_{Reg} = |\partial_x \tilde{d}_t| e^{-|\partial_x x_t|} + |\partial_y \tilde{d}_t| e^{-|\partial_y x_t|} \tag{5}$$

This loss function is used to smooth out the depth gradients by penalizing the large gradients especially when image has smaller gradients. This helps the depth maps follow the image structure.

2.4 Optical Flow Estimation

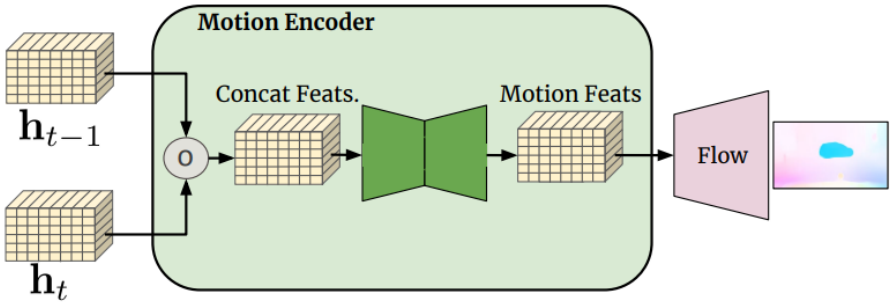

Figure 3: Optical Flow Estimation.

The optical flow, which represents the movement of each pixel between two frames, is also estimated from the motion features h_{motion}. The flow decoder produces a two-channel representation F^t_{t-1}, which contains the vertical and horizontal displacements of pixels:

$$F^t_{t-1} = f_{\text{flow}}(h_{\text{motion}}) \tag{6}$$

The optical flow estimation model uses the same motion encoder as the ego motion estimation model to extract the motion related features. The decoder in this model is based on FlowNet2 [4]. The decoder take the motion features as input and passes them through a series of convolutional layers. The decoder progressively reduces the feature dimensions from the initial 128-channel input to 32 channels and finally to a 2-channel output, which corresponds to the horizontal and vertical components of the optical flow.

To ensure that the predicted optical flow aligns with the original image resolution, the model uses an upsampling step. Specifically, after the flow is predicted from the motion features, it is passed through a bilinear interpolation function to upsample the flow map to the desired output size, which in our ase is the size of the orginal images. This process produces a refined optical flow map that represents the pixel-wise movement between the input images. This model is useful for tasks such as motion estimation, object tracking, and scene understanding, where accurate pixel-level movement information is crucial.

3 Experiments

3.1 Datasets

The dataset used for the experimentation was the CityScapes Dataset [1]. This is a dataset widely used for autonomous driving tasks. It contains sequences with the point-of-view being a moving vehicle. The sequences are divided by cities across Germany, with different cities being part of the train, test and validation splits.

Each sequence of the dataset contains 30 frames from the camera. Frame 20 includes the segmentation labels for the dataset, but for the scope of this model and experiment, the segmentation labels were not relevant. The size of the dataset was a useful factor in the experimentation as it has 5000 sequences across all the train, test and validation splits.

Initially, the dataset was divided into splits containing sub-folders for the cities the sequences belonged to. The dataset was divided into sub-folders, where each folder contained a sequence. This was done to make the the visualization and implementation logic for the models simpler, as they would not have to create the sequences, before processing them.

3.2 Evaluation Metrics

- **Optical Flow Metrics:**
  - **Mean Squared Error (MSE)**: Measures the average squared difference between the estimated flow and the ground truth.
  - **1-Pixel Error (1PE)**: Measures the percentage of pixels where the estimated flow deviates by more than 1 pixel.
  - **3-Pixel Error (3PE)**: Measures the percentage of pixels where the estimated flow deviates by more than 3 pixels.
- **Depth Estimation Metrics:**
  - **Root Mean Squared Error (RMSE)**: Evaluates the discrepancy between predicted and actual depth values.
  - **Accuracy** (< 1.25, $< 1.25^2$): Measures the accuracy of depth predictions within specific thresholds, indicating how close the predicted depth is to the true value.
- **Qualitative Evaluation:**
  - **Visualization**: Depth maps, optical flow, and camera poses are visualized, including the generation of images and GIFs.
  - **Object-Motion Flow**: Computes and visualizes object-motion flow, defined as the difference between full flow and rigid flow.
- **Evaluation Setup:**

– Quantitative evaluation is conducted on frame 20 of every validation sequence, using the same image size as during training.

3.3 Data Augmentation

Data augmentations were a critical part of the training experiments as they contribute to a model's robustness. We used a color jitter and a normalization filter with mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]. The normalization filter was critical in the experimentation as the depth and flow estimation mappings expected pixels in the range [-1, 1]. The validation set used the same augmentations as the training set.

3.4 Training

The project overall is structured in a way that all models have to be trained together as shown in figure 1, as the models are connected to provide the results. All models are trained using resized images of size 256 x 512, with a batch size of 16. The validation set used the same configuration was used for the validation set. A learning rate of 0.0001 was used for the model. Along with that a step learning rate scheduler was used with a step size of 10.

The models are trained in a self-supervised manner, where the objective is to perform novel view synthesis. Given the estimated camera motion C_{t-1}^{t} and depth map d_t, the source frame x_{t-1} can be warped into the target frame x_t. The estimated optical flow F_{t-1}^{t} is also used to warp pixels between frames:

$$\hat{x}_t = \text{Warp}(x_{t-1}, d_t, C_{t-1}^{t}) \tag{7}$$

The loss function is composed of a photometric loss, which measures the similarity between the synthesized image $\hat{x}_t$ and the target image x_t, as well as regularization terms to ensure smoothness in depth and flow estimations.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{photo}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth\_reg}} + \lambda_{\text{flow}}\mathcal{L}_{\text{flow\_reg}} \tag{8}$$

4 Results

The results of the models are shown in the following diagrams. The motion encoder were successfully able to learn the motion features for the depth and ego-motion estimation as the flow estimation delivered acceptable results. However, the decoding was a bit more challenging, especially for the ego-motion depth warp. The flow estimation was successfully able to decode the motion features and create a flow map by detecting major objects in the image sequences.

Figure 4 shows the output of the depth maps from the depth decoder. Figure 5 shows the final result of the optical flow, and figure 6 shows the final depth warp of the image.

The depth warp however, was not able to detect the object boundaries and thus the depth of different objects were not completely learnt. The depths maps

Model	Evaluation Metric	Value
Depth	Root Mean Squared Error	0.6283945179036758
	Accuracy(<1.25)	0.9999923378612862
	Accuracy($<1.25^2$)	0.999996529116651
Flow	Mean Squared Error	33.7577706957995361
	1-Pixel Error	1.0
	3-Pixel Error	1.0

Table 1: Training Results

however performed better when a sigmoid function was added to the model, it performed significantly better, where closer objects were depicted, with blurred boundaries.

The results of our experiments are summarized in table 1.

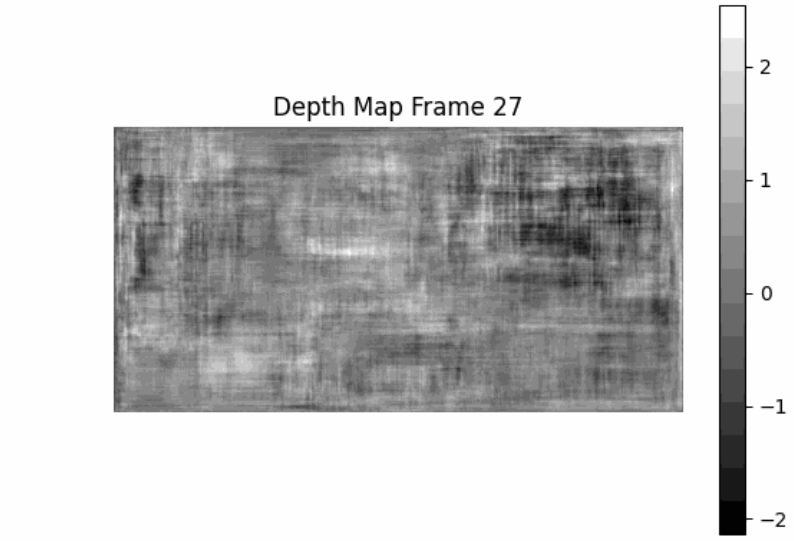

Figure 4: Depiction of the last frame of the generated of the Depth Estimation Map .

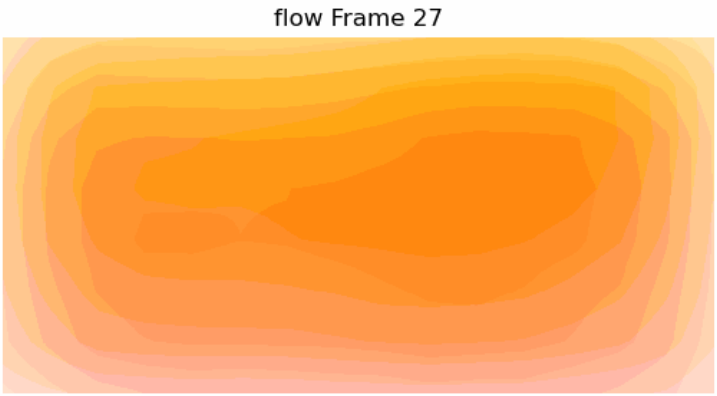

Figure 5: Depiction of the last frame of the generated of the Optical flow Model.

5 Conclusion

This project successfully demonstrated the use of self-supervised learning in the estimation of depth and motion in automotive scenes. We developed an effective model to predict depth, ego-motion, and optical flow by leveraging self-supervised training to handle costly depth annotations and sensor errors. The proposed model is based on an architecture composed of separate image and motion encoders paired with per-task decoders to estimate different scene attributes.

We performed an end-to-end training pipeline for the novel view synthesis, leveraging the Cityscapes dataset by predicting camera transformations and pixel displacements. Our approach was based on synthesizing the target images from the source images and could predict depth and flow very accurately using photometric losses and smooth regularizations. Moreover, this model has shown quantitative and qualitative evaluation effectively in visualizing depth, optical flow, and camera poses.

The experiments and extensive results presented in this paper certainly attest to the feasibility and effectiveness of self-supervised techniques for scene

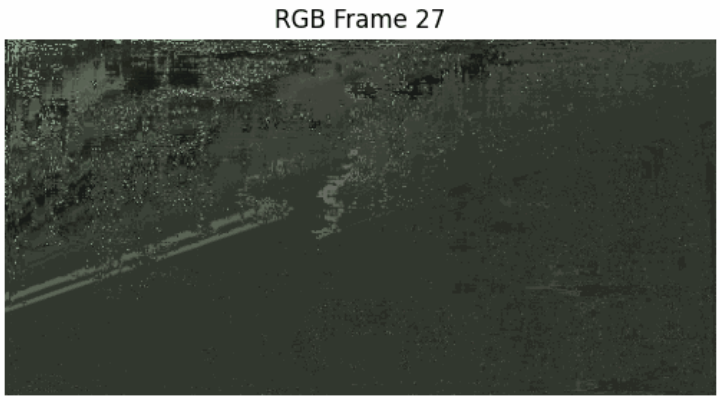

Figure 6: Depiction of the last frame of the generated of the Warp Depth Map.

understanding within autonomous driving contexts. It solves some of the most important challenges inherent in expensive annotations and sensor errors to further the techniques of depth and motion estimation, thus allowing the techniques to become more available and adaptable in real-life scenarios. Further work may be directed towards optimizing architectural design and alternative training schedules, which will allow for more robustness improvements to be made in the model and improve performance on dynamic scenes.

6 Contributions

The code for model architecture were mostly written by Ibrahim and Mridul. Dataloaders were created by Mansi. Discussions and brainstorming for ideas was done regularly (at least twice a week) in which all three of us participated. Some of the code was jointly written by the three of us. Conducting the experiments and creation of the report were done together by the three of us. Generation of figures in the report are done by Mansi.

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[2] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation, 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks, 2016.

[5] Angel Villar-Corrales, Moritz Austermann, and Sven Behnke. Mcds-vss: Moving camera dynamic scene video semantic segmentation by filtering with self-supervised geometry and motion, 2024.